Submission Instructions Please submit a zip file containing two files. One should be a document in a common document format (doc, xls, txt, pdf, pages) containing your solution to questions 1 and 2. The second file should be a python script that answers question 3.

Give document 1 an informative name such as:

`your-uwnetid-tcss501-hw01.docx`

Give your python script this exact (all lower case) filename.

`palindrome.py`

There is no requirement for the name of the zip file.

Sorting Computational Complexity...

Question 1: 5pts Sort the following growth functions $T(n)$ in to the order of ever increasing time complexity $O(f(n))$. Note: There may be some with equal complexity classes. In those cases, indicate the ties by putting them on the same line. If it helps, try plotting the functions to observe their growth for high values of n

1. $n^{3/2}$
2. $100n$
3. $4n^2$
4. $\log_2 n$
5. 3.2^n
6. $0.0001n!$
7. $n * \log_2 n$
8. $5n^2 + 10n$
9. $1n^3 + 5n - 100$
10. $\sqrt{n}$

Question 2: 5pts Consider the algorithm below for calculating summary statistics for a given array. Write determine the growth function $T(n)$ including each labeled section of code. And then combine/compose the sections to determine $O(f(n))$. Your answer should be in the form of
$T(n) = ...$the growth function$... = O($complexity class$)$

```
def summary_stats(n):
    """ First calculates the median, and then the average.
    :param n: An array of numeric values.
    :return: A tuple of (median,average)
    """

    # SEGMENT 1
    a_len = len(n)
    if a_len == 1:  # TRIVIAL CASE WHERE N IS A SINGLE VALUE
        return n[0], n[0]

    # SEGMENT 2
    a = merge_sort(n)  # USE WHAT YOU KNOW ABOUT TIME COMPLEXITY OF MERGE SORT FOR THIS SEGMENT

    # SEGMENT 3
    split = a_len // 2
    median: int = 0
    if a_len % 2 == 0:  # is even
        median = (a[split - 1] + a[split]) / 2
    else:
        median = a[split]

    # SEGMENT 4
    running_sum = 0
    for i in range(0, a_len):
        running_sum += a[i]

    mean = running_sum / a_len

    return (median, mean)


l = [1,7,4,2,1,4,3,6,7,3,12,40]
med, avg = summary_stats(l)
```

Question 3: 5pts Write an algorithm to determine whether or not a given string is a palindrome (a string that is equivalent when written both forward and backward, for example 'racecar' is a palindrome, as is "a man a plan a canal panama".

The implementation should be case agnostic, meaning 'a=A', or put another way 'Racecar' is still a palindrome.

Your implementation needn't remove punctuation. It is okay that 'A man a plan a canal, Panama' is not considered a palindrome but 'A man a Plan a Canal Panama' is.

Note, do NOT use build in string functions such as .reverse() to solve this problem.

```
return the_string == the_string.reverse()
```

will be given no credit.

Use the following definition for your function.

function:

```
    def is_palindrome(the_string):
    """
       Evaluates a given string and determines whether or not it is a palindrome.
       :param the_string: The string to evaluate.
       :returns: True when the string is a palindrome, False otherwise.
    """
```

Submit the function as a single python file with a comment header that includes your name and uwnetid. Also include the time complexity in terms of the input length (N) of the evaluated string. example:

```
"""
  Name: Kevin Anderson
  UWNetId: k3a
  TimeComplexity = O(xxxxxx)
"""
```